MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

| **Module Information**  **معلومات المادة الدراسية** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Module Title** | Object Oriented Programming | | | | **Module Delivery** | | |
| **Module Type** | Core | | | | * **☒ Theory** * **☐ Lecture** * **☒ Lab** * **☐ Tutorial** * **☐ Practical** * **☐ Seminar** | | |
| **Module Code** | ITC320070 | | | |
| **ECTS Credits** | 6.00 | | | |
| **SWL (hr/sem)** | 150 | | | |
| **Module Level** | | 2 | **Semester of Delivery** | | | | 3 |
| **Administering Department** | | BID | **College** | BMIC | | | |
| **Module Leader** | Mohammed Fadhil | | **e-mail** | mfadhil@uoitc.edu.iq | | | |
| **Module Leader’s Acad. Title** | | Lecturer | **Module Leader’s Qualification** | | | | Ph.D. |
| **Module Tutor** |  | | **e-mail** |  | | | |
| **Peer Reviewer Name** | | jwan k alwan | **e-mail** | jwanism@uoitc.edu.iq | | | |
| **Scientific Committee Approval Date** | | 18/06/2023 | **Version Number** | | | 1.0 | |

| **Relation with other Modules**  **العلاقة مع المواد الدراسية الأخرى** | | | |
| --- | --- | --- | --- |
| **Prerequisite module** | Computer Programming II / BMI121 | **Semester** | 2 |
| **Co-requisites module** | None | **Semester** |  |

| **Module Aims, Learning Outcomes and Indicative Contents**  **أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية** | |
| --- | --- |
| **Module Aims**  **أهداف المادة الدراسية** | 1. To learn how to analyze and design software systems using object-oriented techniques. 2. To understand the importance of objects, classes, and the relationships between them. 3. To learn the fundamental principles of OOP, including encapsulation, inheritance, and polymorphism. 4. To learn how to design and implement modular, reusable, and maintainable code using object-oriented techniques. 5. To apply OOP principles and concepts in practical software development. |
| **Module Learning Outcomes**  **مخرجات التعلم للمادة الدراسية** | 1. Understand Software development patterns. 2. Analyze and design software systems using object-oriented techniques. 3. Understand the importance of objects, classes, and their relationships. 4. Implement classes, methods, and constructors in object-oriented programming languages. 5. Constructors and destructors. 6. Apply the fundamental principles of OOP: encapsulation and abstraction. 7. Design modular, reusable, and maintainable code using object-oriented techniques. 8. Apply the fundamental principles of OOP: inheritance, and polymorphism. 9. Demonstrate proficiency in creating and manipulating objects. 10. Utilize inheritance and polymorphism to establish relationships between classes. 11. Apply OOP principles and concepts in practical software development. 12. Apply code organization, separation of concerns, and abstraction for better software design. |
| **Indicative Contents**  **المحتويات الإرشادية** | Indicative content includes the following.  Analyzing and Designing Software Systems   * Introduction to software analysis and design * Understanding requirements and problem domain analysis * Object-oriented modeling techniques   Importance of Objects, Classes, and Relationships   * Introduction to objects and classes * Understanding relationships between objects and classes * Composition, inheritance, and association   Fundamental Principles of OOP   * Introduction to object-oriented programming * Encapsulation and data hiding * Inheritance and polymorphism * Abstraction and interfaces   Designing Modular, Reusable, and Maintainable Code   * Principles of code organization and modular design * Design patterns for code reusability * Creating reusable components and libraries   Applying OOP Principles in Software Development   * Language-specific features and syntax * Creating and manipulating objects * Implementing classes, methods, and constructors * Inheritance and polymorphism in practice |

| **Learning and Teaching Strategies**  **استراتيجيات التعلم والتعليم** | |
| --- | --- |
| **Strategies** | The module strategy for Object-Oriented Programming focuses on a combination of theoretical understanding and practical application of object-oriented concepts and principles. The strategy aims to engage students in active learning, fostering their ability to analyze, design, and implement software systems using object-oriented techniques. |

| **Student Workload (SWL)**  **الحمل الدراسي للطالب** | | | |
| --- | --- | --- | --- |
| **Structured SWL (h/sem)**  **الحمل الدراسي المنتظم للطالب خلال الفصل** | 63 | **Structured SWL (h/w)**  **الحمل الدراسي المنتظم للطالب أسبوعيا** | 4 |
| **Unstructured SWL (h/sem)**  **الحمل الدراسي غير المنتظم للطالب خلال الفصل** | 87 | **Unstructured SWL (h/w)**  **الحمل الدراسي غير المنتظم للطالب أسبوعيا** | 6 |
| **Total SWL (h/sem)**  **الحمل الدراسي الكلي للطالب خلال الفصل** | 150 | | |

| **Module Evaluation**  **تقييم المادة الدراسية** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **As** | | **Time/Number** | **Weight (Marks)** | **Week Due** | **Relevant Learning Outcome** |
| **Formative assessment** | **Quizzes** | 3 | 10% (10) | 3, 10, 13 | LO #1, 2, 10 and 11 |
| **Assignments** | 2 | 10% (10) | 2, 12 | LO # 3, 4, 6 and 7 |
| **Projects / Lab.** | 1 | 10% (10) | Continuous |  |
| **Report** | 1 | 10% (10) | 15 | LO # 5, 8 and 10 |
| **Summative assessment** | **Midterm Exam** | 2 hr | 10% (10) | 7,14 | LO # 1-7 |
| **Final Exam** | 3hr | 50% (50) | 16 | All |
| **Total assessment** | | | 100% (100 Marks) |  |  |

| **Delivery Plan (Weekly Syllabus)**  **المنهاج الاسبوعي النظري** | |
| --- | --- |
| **Week** | **Material Covered** |
| **Week 1** | Introduction - Software Development |
| **Week 2** | Programming Paradigms |
| **Week 3** | Introduction to OOP and its need |
| **Week 4** | Software Analysis and Design using OOP Techniques |
| **Week 5** | Objects, Classes, and Relationships |
| **Week 6** | Attributes and Methods |
| **Week 7** | 1st Exam |
| **Week 8** | Encapsulation and Data Hiding in OOP |
| **Week 9** | Abstraction and Interfaces in OOP |
| **Week 10** | Inheritance and Access Modifiers |
| **Week 11** | OO Concepts: Inheritance and Polymorphism |
| **Week 12** | Documentation in software development |
| **Week 13** | Creating Reusable Components and Libraries |
| **Week 14** | Tutorial for the course topics |
| **Week 15** | 2nd Exam |

| **Delivery Plan (Weekly Lab. Syllabus)**  **المنهاج الاسبوعي للمختبر** | |
| --- | --- |
| **Week** | **Material Covered** |
| **Week 1** | Lab 1: Introduction to Python Syntax |
| **Week 2-3** | Lab 2-3: Build first class |
| **Week 4** | Lab 4: Class Data and methods |
| **Week 5** | Lab 5: Object Data and methods |
| **Week 6-7** | Lab 6-7: Encapsulation practical examples |
| **Week 8** | Lab 8: Abstraction practical examples |
| **Week 9-10** | Lab 9-10: Inheritance practical examples |
| **Week 11-12** | Lab 11-12: Polymorphism practical examples |
| **Week 13-14** | Lab 13-14: Documentations and libraries |
| **Week 15** | Presentations and discussions |

| **Learning and Teaching Resources**  **مصادر التعلم والتدريس** | | |
| --- | --- | --- |
|  | **Text** | **Available in the Library?** |
| **Required Texts** | Python 3 Object-Oriented Programming by Dusty Phillips | No |
| **Recommended Texts** | Python Programming: An Introduction to Computer Science  by John M. Zelle, Ph.D. | No |
| **Websites** |  | |

| **Grading Scheme**  **مخطط الدرجات** | | | | |
| --- | --- | --- | --- | --- |
| **Group** | **Grade** | التقدير | **Marks (%)** | **Definition** |
| **Success Group**  **(50 - 100)** | **A -** Excellent | **امتياز** | 90 - 100 | Outstanding Performance |
| **B -** Very Good | **جيد جدا** | 80 - 89 | Above average with some errors |
| **C -** Good | **جيد** | 70 - 79 | Sound work with notable errors |
| **D -** Satisfactory | **متوسط** | 60 - 69 | Fair but with major shortcomings |
| **E -** Sufficient | **مقبول** | 50 - 59 | Work meets minimum criteria |
| **Fail Group**  **(0 – 49)** | **FX –** Fail | **راسب (قيد المعالجة)** | (45-49) | More work required but credit awarded |
| **F –** Fail | **راسب** | (0-44) | Considerable amount of work required |
|  |  |  |  |  |
| **Note:** Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above. | | | | |